

A Convex Feasibility Approach to Anytime Model Predictive Control

Alberto Bemporad, Daniele Bernardini, Panagiotis Patrinos

Abstract—This paper proposes to decouple performance optimization and enforcement of asymptotic convergence in Model Predictive Control (MPC) so that convergence to a given terminal set is achieved independently of how much performance is optimized at each sampling step. By embedding an explicit decreasing condition in the MPC constraints and thanks to a novel and very easy-to-implement convex feasibility solver proposed in the paper, it is possible to run an outer performance optimization algorithm on top of the feasibility solver and optimize for an amount of time that depends on the available CPU resources within the current sampling step (possibly going open-loop at a given sampling step in the extreme case no resources are available) and still guarantee convergence to the terminal set. While the MPC setup and the solver proposed in the paper can deal with quite general classes of functions, we highlight the synthesis method and show numerical results in case of linear MPC and ellipsoidal and polyhedral terminal sets.

I. INTRODUCTION

Model Predictive Control (MPC) is a well known advanced control approach in industry for its capability of optimizing closed-loop performance subject to operating constraints on input and output variables [1]–[3]. In recent years, MPC has become very attractive also in fast-sampling applications with stringent real-time requirements, such as those arising in the automotive and aerospace industries. Such requirements posed a research challenge for developing optimization algorithms, and in particular Quadratic Programming (QP) solvers, that enable the use of MPC in commercial products. In particular, an embedded optimization solver must be fast, simple to code and test, require little memory, and have good worst-case estimates of its execution time.

To cope with such requirements, multiparametric QP was proposed in [4] to pre-solve the QP off-line, therefore converting the MPC law into a continuous and piecewise affine function of the state vector. The main drawback of explicit MPC is that it is limited to relatively small problems and to linear time-invariant (LTI) systems.

On-line optimization methods like active-set methods [5]–[7], interior-point methods [8]–[10], and dual piecewise smooth Newton methods [11] can be very effective in speed, but their worst-case CPU time can be hard to estimate in a non-conservative way. For accelerated dual gradient-projection methods [12], good bounds on the worst-case execution time were provided [13], [14], although the methods act on the dual QP problem, and therefore can lead to infeasible solutions if the execution is interrupted.

On the other hand, in real-time control platforms the time allotted for the MPC controller to run is often not enough to cover the worst-case execution time, and other higher-priority

tasks may even preempt its full execution. Driven by such real-time constraints, *anytime control algorithms* were developed in [15] with the idea of storing a set of control laws, each one of different complexity and closed-loop performance, and execute the one whose complexity is compatible with the current available CPU resources.

In this paper we propose instead an MPC approach based on *anytime optimization*, with a novel convex optimization algorithm that recursively finds feasible solutions of decreasing level of suboptimality, depending on the computation power available within the sampling step. We first prove a rather general recursive feasibility and convergence result of MPC based on stability constraints that artificially impose a certain Lyapunov function to be decreasing [16], [17], where this function might be totally decoupled from the value function typically considered for assessing asymptotic convergence [10], [18]. Moreover, in this paper we focus on *convergence to a set* around an equilibrium rather than to an equilibrium state, as in practical applications is often sufficient to track a set-point within a given tolerance [19]. In addition, contrarily to [10] that guarantees feasibility in real-time through a warm-starting technique in combination with robust MPC design, we provide an approach based on an original method for solving unconstrained problems, which is used for finding a feasible solution to a set of convex constraints. This method is very efficient in speed and easy to code, and can be run multiple times to approach an optimal solution, depending on the available CPU time.

The paper is organized as follows. Section II defines the main MPC setup and states recursive feasibility and convergence results. Section III presents the new convex feasibility and optimization algorithm setup and shows its properties. Section IV proposes two ways of synthesizing a proper terminal set and stability constraints for linear systems subject to linear constraints on inputs and outputs, and Section V shows numerical evidence of the advantages of the proposed approach.

A. Notation

The sets of real and nonnegative integer numbers are denoted by \mathbb{R} , \mathbb{N} , respectively. For a vector $x \in \mathbb{R}^n$, x_i denotes the i -th entry of x , and the expression $x > 0$ means that $x_i > 0$, $\forall i = 1, \dots, n$. For a matrix $A \in \mathbb{R}^{n \times m}$, A_i denotes the i -th row of A , and $A > 0$ positive definiteness of A . Given a scalar τ , τ_+ denotes $\max\{\tau, 0\}$; for a vector $x \in \mathbb{R}^m$, x_+ is the vector whose coordinates are $(x_+)_i = \max\{x_i, 0\}$, $\forall i = 1 \dots, m$.

II. FEASIBILITY-BASED MPC

Consider the problem of steering the system

$$x(t+1) = \alpha(x(t), u(t)) \quad (1)$$

to a target set $\mathcal{S} \subseteq \mathbb{R}^n$ while satisfying the constraints

$$g(x(t), u(t)) \leq 0 \quad (2)$$

for all $t \in \mathbb{N}$, where $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $\alpha : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n$ and $g : \mathbb{R}^{n+m} \rightarrow \mathbb{R}$. We represent the target set as

$$\mathcal{S} \triangleq \{x \in \mathbb{R}^n : f(x) \leq 0\} \quad (3)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and \mathcal{S} is constrained controlled invariant with respect to (2), in accordance with the following definition.

Definition 1: A set $\mathcal{S} \subseteq \mathbb{R}^n$ is *constrained controlled invariant* if for all $x \in \mathcal{S}$ there exists $u \in \mathbb{R}^m$ such that $g(x, u) \leq 0$, $\alpha(x, u) \in \mathcal{S}$.

Note that in (2)–(3) we are assuming scalar functions f , g without loss of generality. In fact, for any vector function $h : \mathbb{R}^{n_1} \rightarrow \mathbb{R}^{n_2}$, the component-wise constraint $h(x) \leq 0$ is equivalent to the constraint $\bar{h}(x) \triangleq \max_{i=1, \dots, n_2} h_i(x) \leq 0$. Note also that, given any set $\mathcal{S} \subseteq \mathbb{R}^n$, a corresponding function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ satisfying (3) can be defined as $f(x) = \gamma_{\mathcal{S}}(x) - 1$, where $\gamma_{\mathcal{S}}(x) = \inf\{\mu : \mu \geq 0, x \in \mu\mathcal{S}\}$ is the Minkowski function of \mathcal{S} .

To solve the stated control problem, we consider the following MPC formulation

$$\min \ell_N(x_N) + \sum_{k=0}^{N-1} \ell_k(x_k, u_k) \quad (4a)$$

$$\text{s.t. } x_0 = x(t) \quad (4b)$$

$$x_{k+1} = \alpha(x_k, u_k), \quad k = 0, 1, \dots, N-1 \quad (4c)$$

$$g(x_k, u_k) \leq 0, \quad k = 0, 1, \dots, N-1 \quad (4d)$$

$$f(x_N) \leq 0 \quad (4e)$$

$$\sum_{k=1}^{N-1} f(x_k)_+ \leq \phi(t-1) \quad (4f)$$

where $\ell_k : \mathbb{R}^{n+m} \rightarrow \mathbb{R}$, $k = 0, \dots, N-1$, $\ell_N : \mathbb{R}^n \rightarrow \mathbb{R}$ are stage and terminal costs, respectively, and $\phi(t)$ is a given scalar, chosen in accordance with the following theorem.

Theorem 2: Let $u(t) = u_0^t$ be the control input applied to the process (1), where $\{u_k^t\}_{k=0}^{N-1}$ is any feasible solution of problem (4) at time t , and the quantity

$$\phi(t-1) \triangleq \sum_{k=1}^{N-1} f(x_k^{t-1})_+ \quad (5)$$

is constructed from the previous feasible solution u_k^{t-1} , x_k^{t-1} of problem (4) at time $t-1$, for all $t \in \mathbb{N}$. If the set \mathcal{S} defined in (3) is constrained controlled invariant and problem (4) is feasible at time $t=0$ for the initial state $x(0)$ and some value $\phi(-1)$, then it is feasible at all time $t \in \mathbb{N}$ and $x(t) \rightarrow \mathcal{S}$ for $t \rightarrow \infty$.

Proof: Let $\{u_k^t\}_{k=0}^{N-1}$ be any feasible solution of problem (4) chosen at time t , and let $\{x_k^t\}_{k=0}^N$ be the corresponding state trajectory. Consider the following candidate feasible solution $\{\bar{u}_k\}_{k=0}^{N-1}$ at time $t+1$, where $\bar{u}_0 = u_1^t$, $\bar{u}_1 = u_2^t$,

\dots , $\bar{u}_{N-2} = u_{N-1}^t$, and \bar{u}_{N-1} such that $g(x_N^t, \bar{u}_{N-1}) \leq 0$, which exists by constrained controlled invariance of \mathcal{S} . Let $\{\bar{x}_k\}_{k=0}^N$ the state trajectory corresponding to $\{\bar{u}_k\}_{k=0}^{N-1}$, with $\bar{x}_0 = x(t+1)$. By construction, $\bar{x}_k = x_{k+1}^t$ for all $k = 0, \dots, N-1$, and hence $g(\bar{x}_k, \bar{u}_k) \leq 0$ for all $k = 0, \dots, N-2$, $f(\bar{x}_{N-1}) \leq 0$. Moreover, by the choice of \bar{u}_{N-1} we have $g(\bar{x}_{N-1}, \bar{u}_{N-1}) \leq 0$ and $f(\bar{x}_N) \leq 0$. Since $u(t) = u_0(t)$, and hence $x_1^t = x(t+1)$, we have

$$\sum_{k=1}^{N-1} f(\bar{x}_k)_+ = \sum_{k=2}^{N-1} f(x_k^t)_+ = \phi(t) - f(x(t+1))_+ \quad (6)$$

so that, since $f(x(t+1))_+ \geq 0$, also the stability constraint (4f) is satisfied. Therefore, problem (4) admits a feasible solution at time $t+1$, and because of (6), whatever is the choice of $\{u_k^{t+1}\}_{k=0}^{N-1}$ at time $t+1$, we have

$$\phi(t+1) \leq \phi(t) - f(x(t+1))_+ \quad (7)$$

This proves that $\lim_{t \rightarrow \infty} \phi(t)$ exists, as ϕ is a monotonically decreasing sequence and lower-bounded by zero, which in turns implies by (7) that $\lim_{t \rightarrow \infty} f(x(t))_+ = 0$. If by contradiction we assume that $x(t) \not\rightarrow \mathcal{S}$ for $t \rightarrow \infty$, then a subsequence $t_h \in \mathbb{N}$, $h \in \mathbb{N}$, and a scalar $\delta > 0$ exist such that $f(x(t_h)) \geq \delta$, $\forall h \in \mathbb{N}$, or equivalently $f(x(t_h))_+ \geq \delta$, which contradicts $\lim_{t \rightarrow \infty} f(x(t))_+ = 0$. ■

Note that the convergence result of Theorem 2 does not involve at all the cost function (4a). Of course, the transient behaviour of the system depends on how close to optimality are the chosen feasible solutions $\{u_k^t\}_{k=0}^{N-1}$ of (4).

While the result of Theorem (2) does not make any assumption on the properties of functions α , f , g , ℓ_k (except for constrained controlled invariance of \mathcal{S}), from now on we will restrict our attention to *convex* functions f , g , ℓ_k and *linear* functions α , i.e., linear models

$$x(t+1) = Ax(t) + Bu(t) \quad (8)$$

in order to solve problem (4) effectively, using the novel algorithm proposed in the next section.

III. CONVEX FEASIBILITY ALGORITHM

Consider the following feasibility problem:

$$\text{Find } x \in C \triangleq \{x \in \mathbb{R}^n : f_i(x) \leq 0, i = 1, \dots, m\}, \quad (9)$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, m$ are convex, twice continuously differentiable functions. Problem (9) can be reformulated as the following unconstrained minimization problem:

$$\min F(x) = \frac{1}{2} \|f(x)_+\|_2^2 \quad (10)$$

where $f(x) = [f_1(x) \dots f_m(x)]$.

Proposition 3: If $C \neq \emptyset$, then $\inf F = 0$ and $\arg \min F = C$.

Clearly, if $\inf F > 0$ then C is empty.

Lemma 4: Function $F(x) = \frac{1}{2} \|f(x)_+\|_2^2$ is convex and continuously differentiable with

$$\nabla F(x) = \nabla f(x)' f(x)_+ = \sum_{i=1}^m f_i(x)_+ \nabla f_i(x) \quad (11)$$

Proof: Function F can be written as

$$F(x) = \frac{1}{2} \sum_{i=1}^m (\max\{f_i(x), 0\})^2 = \frac{1}{2} \sum_{i=1}^m q(\psi_i(x)),$$

where $\mathbb{R} \ni z \mapsto q(z) = z^2$ and $\mathbb{R} \ni z \mapsto \psi_i(x) = \max\{f_i(x), 0\}$, $q \circ \psi$ is convex since q is convex and non-decreasing when its argument is nonnegative and ψ_i is convex (as the pointwise maximum of the convex function f_i and 0) and nonnegative. Therefore F is convex as the sum of convex functions. On the other hand, $F(x) = \sum_{i=1}^m \varphi(f_i(x))$ where $\mathbb{R} \ni z \mapsto \varphi(z) = \frac{1}{2}z_+^2$. Since φ is continuously differentiable with $\frac{d\varphi(z)}{dz} = (z)_+$, continuous differentiability of F and formula (11) readily follow. ■

Our ultimate goal is to devise Newton-like methods for solving the unconstrained problem (10) and thus (9). Function F is \mathcal{C}^1 but not \mathcal{C}^2 , however its gradient $\nabla F(x)$ is a piecewise smooth mapping, in the sense that for any $x \in \mathbb{R}^n$ we have

$$\nabla F(x) \in \{\nabla F_I(x)\}_{I \in \mathcal{I}}, \quad (12)$$

where \mathcal{I} is the collection of all subsets $I \subseteq \{1, \dots, m\}$ for which there exists a $x \in \mathbb{R}^n$ such that $f_i(x) \geq 0$, $i \in I$ and $f_i(x) < 0$, $i \notin I$, whereas $\nabla F_I(x) = \sum_{i \in I} f_i(x) \nabla f_i(x)$. The pieces of ∇F are smooth with Jacobian given by

$$\nabla^2 F_I(x) = \sum_{i \in I} \nabla f_i(x) \nabla f_i(x)' + f_i(x) \nabla^2 f_i(x). \quad (13)$$

For any $x \in \mathbb{R}^n$ let $I(x) = \{i \in \{1, \dots, m\} : f_i(x) \geq 0\}$. Then the matrix $\nabla^2 F_{I(x)}$ will serve as a generalized Hessian of F at x , furnishing a second-order approximation similar to the one provided by the classical Hessian for \mathcal{C}^2 functions. Another idea, stemming from Gauss-Newton methods for solving least-squares problems, would be to use as a generalized Hessian the matrix $\sum_{i \in I(x)} \nabla f_i(x) \nabla f_i(x)'$, which results by omitting second-order terms $\nabla^2 f_i(x) f_i(x)$ from $\nabla^2 F_{I(x)}$. This choice saves us from computing the Hessians of f_i , $i \in I(x)$ (notice that in case of $f_i(x) = a_i'x - b_i$ this makes no difference). However, this is a good choice only if we know that C is nonempty. In this case for any $\bar{x} \in C$ we have $f_i(\bar{x}) = 0$ for $i \in I(\bar{x})$ and the term $\nabla^2 f_i(\bar{x}) f_i(\bar{x})$ vanishes.

Algorithm 1 is a regularized piecewise smooth Newton method with line search. Its convergence properties can be inferred as a special case of [20], [21]. Specifically, every accumulation point of the sequence generated is a stationary point of F , and if $\nabla^2 F_{I(x^*)}$ is nonsingular then the convergence rate is quadratic.

A. Feasibility-based optimization

Problems involving constrained minimization of a \mathcal{C}^2 convex function can be attacked by solving a sequence of feasibility problems. Specifically consider the problem

$$\min f_0(x) \quad (16a)$$

$$\text{s.t. } x \in C \quad (16b)$$

where $C \subseteq \mathbb{R}^m$ is a closed convex set described as in (9). Let $f^* = \inf_{x \in C} f_0(x)$ and $X^* = \arg \min_{x \in C} f_0(x)$. We assume

Algorithm 1: [empty, x]=PSN_FEAS(C)

Input: $\sigma \in (0, 1/2)$, $\zeta \in (0, 1)$, $x^0 \in \mathbb{R}^n$, $k = 0$

1 **if** $F(x^k) = 0$ **then**
2 | empty ← false, $x \leftarrow x^k$; **exit**
3 **else if** $\|\nabla F(x^k)\| = 0$ **then**
4 | empty = true; **exit**
5 **end**
6 Compute $d^k \in \mathbb{R}^n$ that solves

$$(\nabla^2 F_{I^k}(x^k) + \delta^k I)d = -\nabla F(x^k), \quad (14)$$

where $I^k = \{i \in [m] \mid f_i(x^k) \geq 0\}$, $\delta^k = \zeta \|\nabla F(x^k)\|$.

7 Compute $\tau^k = \max\{2^{-i} \mid i = 0, 1, 2, \dots\}$ such that

$$F(x^k + \tau^k d^k) \leq F(x^k) + \sigma \tau^k \nabla F(x^k)' d^k. \quad (15)$$

8 $x^{k+1} \leftarrow x^k + \tau^k d^k$

9 $k \leftarrow k + 1$ and go to Step 1.

that the set of optimal solutions of (16) is nonempty. We have that

$$f^* = \inf\{t : x \in S(t)\}, \quad X^* = S(f^*)$$

where

$$S(t) = \{x \in C : f_0(x) \leq t\} \quad (17)$$

is the lower level set of f_0 over C . Obviously we have that $t \geq f_*$ if and only if $S(t)$ is nonempty. This suggests that we can test whether a given t is smaller or larger than f_* by solving the feasibility problem of finding $x \in S(t)$

$$\varphi(t) = \inf_{x \in \mathbb{R}^n} \gamma(t, x) \triangleq \frac{1}{2}(f_0(x) - t)_+^2 + \frac{1}{2} \sum_{i=1}^m (f_i(x)_+)^2 \quad (18)$$

using Algorithm 1. The following proposition, whose proof is omitted here for lack of space, proves some interesting properties enjoyed by function φ .

Proposition 5: Assume problem (16) admits an optimal solution and let function φ be defined as in (18).

- (i) φ is real-valued with $\varphi(t) > 0$ for $t < f^*$, whereas $\varphi(t) = 0$ for $t \geq f^*$,
- (ii) φ is convex and continuously differentiable with

$$\frac{d\varphi(t)}{dt} = \nabla_t \gamma(t, x_t) = -(f_0(x_t) - t)_+, \quad (19)$$

where $x_t \in \arg \min_{x \in \mathbb{R}^n} \gamma(x, t)$.

- (iii) $\frac{d\varphi(t)}{dt} < 0$ for $t < f^*$, whereas $\frac{d\varphi(t)}{dt} = 0$ for $t \geq f^*$.

Proposition 5(i) shows that f^* is the left endpoint of the halfline $\{t \in \mathbb{R} : \varphi(t) = 0\}$. Therefore problem (16) has been reduced to finding the leftmost zero of the one-dimensional, monotone decreasing function $\varphi : \mathbb{R} \rightarrow \mathbb{R}$. One way to find f^* is to apply bisection to φ . Starting from an initial closed interval $[t_-, t_+]$ with $t_- \leq f^* \leq t_+$ we pick the midpoint $t = (t_- + t_+)/2$ and try to determine if the level set $S(t)$ is nonempty, by solving (18) using Algorithm 1 (of course we could apply any other algorithm for unconstrained \mathcal{C}^1 optimization). If $\varphi(t) = 0$ then $S(t)$ is nonempty and this means that the corresponding $x_t \in \arg \min_{x \in \mathbb{R}^n} \gamma(x, t)$ is

feasible for (16), therefore $t \geq f^*$ and the new interval is reduced to $[t_-, t]$. In the case where $\varphi(t) > 0$, $S(t)$ is empty, meaning $t \leq f^*$, so the new interval becomes $[t, t_+]$.

1) *Strengthening the lower bound*: Overall, the bisection algorithm maintains a lower and upper bound for f^* . Since the interval is halved at every bisection step, we obtain the standard linear convergence for bisection, that is, the algorithm stops after at most $\left\lceil \log_2 \left(\frac{t_+ - t_-}{\epsilon} \right) \right\rceil$ steps, where $\epsilon > 0$ is the desired optimality threshold. However, with almost no extra effort we can do much better in practice.

Suppose that $t < f^*$. The optimality condition for problem (18) is $\nabla_x \gamma(x_t, t) = 0$ or

$$(f_0(x_t) - t)_+ \nabla f_0(x_t) + \sum_{i=1}^m (f_i(x_t))_+ \nabla f_i(x_t) = 0. \quad (20)$$

From Proposition 5(iii), we have that $f(x_t) > t$. Dividing by $f_0(x_t) - t$ in (20) and letting $\mu_i = \frac{(f_i(x_t))_+}{f_0(x_t) - t}$, $i = 1, \dots, m$, we obtain

$$\nabla f_0(x_t) + \sum_{i=1}^m \mu_i \nabla f_i(x_t) = 0.$$

Since $\mu \geq 0$, it follows that μ is a dual feasible vector, therefore

$$f_0(x_t) + \sum_{i=1}^m \mu_i f_i(x_t) \leq f^*. \quad (21)$$

Hence $t_D = f_0(x_t) + \sum_{i=1}^m \mu_i f_i(x_t)$ provides a lower bound on f^* . Since $t_D - t = f_0(x_t) + \sum_{i=1}^m \mu_i f_i(x_t) > 0$, t_D is indeed a tighter lower bound to f^* than t .

2) *Strengthening the upper bound*: When $t \geq f^*$, due to 5(iii), we have that $f(x_t) \leq t$, $x_t \in \arg \min_{x \in \mathbb{R}^n} \gamma(x, t)$. Therefore, if $\varphi(t) = 0$, x_t is a feasible vector for problem (16) and $f(x_t)$ is a tighter upper bound to f^* than t .

In fact, we can do even better. At every step of bisection we have at our disposal two vectors x_F and x_I , corresponding to the upper and lower bounds on f^* , respectively. Vector x_F is feasible, i.e., $f_i(x_F) \leq 0$, $i = 1, \dots, m$, while x_I is infeasible, i.e., $f_i(x_I) > 0$ for at least one i and $f_0(x_I) < f_0(x_F)$ (this follows directly from (21)). Invoking a result by Bertsekas [22, Proposition 2] we have that

$$f_* \leq \frac{\Gamma}{\Gamma + 1} f_0(x_F) + \frac{1}{\Gamma + 1} f_0(x_I) \leq f_0(x_F),$$

where $\Gamma = \inf\{\gamma \geq 0 \mid f_i(x_I) \leq -\gamma f_i(x_F), i = 1, \dots, m\}$ [22, Proposition 2]. The bound is nontrivial, i.e., the rightmost inequality is strict, when $\Gamma < \infty$, which holds if and only if $f_i(x_I) \leq 0$ for all i with $f_i(x_F) = 0$. In that case we have $\Gamma = \max_{\{i \mid f_i(x_F) < 0\}} \frac{f_i(x_I)}{-f_i(x_F)}$.

The proposed improved bisection method is summarized in Algorithm 2.

3) *Equality constraints*: The approach can be immediately extended to handle linear equality constraints

$$c'_j x = d_j, \quad j = 1, \dots, m_e$$

in (9), by simply adding the term $\frac{1}{2} \sum_{j=1}^{m_e} (c'_j x - d_j)^2$ in (10) and to $\varphi(t)$ in (18), respectively.

Algorithm 2: PSN_OPT(f_0, C)

Input: accuracy ϵ , $x_F \in C$, $t_+ = f_0(x_F)$, $x_I \notin C$,
 $t_- = f_0(x_I)$ with $t_- \leq f_*$

Output: $x_F \in C$ with $f(x_F) - f^* \leq \epsilon$

```

1 while  $t_+ - t_- > \epsilon$  do
2    $t \leftarrow (t_- + t_+)/2$ 
3   Call  $[\text{empty}, x_t] = \text{PSN\_FEAS}(S(t))$  (Algorithm 1)
4   if  $\text{empty} = \text{true}$  then
5      $x_I \leftarrow x_t$ 
6      $t_- \leftarrow f_0(x_I) + \sum_{i=1}^m \mu_i f_i(x_I)$ , where
7        $\mu_i = \frac{(f_i(x_I))_+}{f_0(x_I) - t_-}$ 
8       if  $f_i(x_I) \leq 0$  for all  $i$  with  $f_i(x_F) = 0$  then
9          $t_+ \leftarrow \frac{\Gamma}{\Gamma+1} f_0(x_F) + \frac{1}{\Gamma+1} f_0(x_I)$ , where
10         $\Gamma = \max_{\{i \mid f_i(x_F) < 0\}} \frac{f_i(x_I)}{-f_i(x_F)}$ 
11      end
12   else
13      $x_F \leftarrow x$ 
14     if  $f_i(x_I) \leq 0$  for all  $i$  with  $f_i(x_F) = 0$  then
15        $t_+ \leftarrow \frac{\Gamma}{\Gamma+1} f_0(x_F) + \frac{1}{\Gamma+1} f_0(x_I)$ , where
16        $\Gamma = \max_{\{i \mid f_i(x_F) < 0\}} \frac{f_i(x_I)}{-f_i(x_F)}$ 
17     else
18        $t_+ \leftarrow f_0(x_F)$ 
19     end
20   end
21 end
```

4) *Determining initial upper and lower bounds*: To determine an initial upper bound t_+ to f^* we can solve the feasibility problem (9) using Algorithm 1. Determining a lower bound is a more delicate issue. If $f_0(x) = (1/2)x'Qx + q'x$ where Q is symmetric positive definite, we can simply determine a lower bound on f^* by computing the unconstrained minimum $x = -Q^{-1}q$. In general, if f_0 is convex and coercive we can find a $x \in \mathbb{R}^n$ such that $\nabla f(x) = 0$. The nonlinear system can be solved by Algorithm 1. In the case of a quadratic program with the cost having a positive semidefinite Hessian a lower bound to f^* can be determined by solving the following convex feasibility problem

$$\text{find } x \in \mathbb{R}^n, \mu \in \mathbb{R}^m, \text{ s.t. } \nabla f_0(x) + \sum_{i=1}^m \mu_i \nabla f_i(x) = 0, \mu \geq 0$$

Then μ is a dual feasible solution and $q = f_0(x) + \sum_{i=1}^m \mu_i f_i(x) \leq f^*$ (even if strong duality does not hold). Another way to determine a lower bound for general convex problems is to find a dual feasible vector, corresponding to the primal feasible vector x_F corresponding to t_+ :

$$\text{find } \mu \in \mathbb{R}^m, \text{ s.t. } \nabla f_0(x_F) + \sum_{i=1}^m \mu_i \nabla f_i(x_F) = 0, \mu \geq 0.$$

Notice that the set of μ satisfying the conditions above is polyhedral.

5) *Special cases*: Algorithm 1 and 2 can be used to solve linear programs (LPs), quadratic programs (QPs), and quadratically-constrained quadratic programs (QCQPs). In this

case, the computations in (12) and (13) require only matrix-vector products.

B. Applicability to feasibility-based MPC

Algorithm 2 requires all the constraints in the inner feasibility problem to be twice differentiable functions. In particular, constraint (4f) is not continuously differentiable because of the max operator, so the above algorithm cannot be directly applied to solve (4). However, we can simply recast the problem by introducing $N-1$ additional variables ϵ_k , $k = 0, \dots, N-1$ and replace (4f) with

$$\epsilon_k \geq f(x_k), \quad k = 1, 2, \dots, N-1 \quad (22a)$$

$$\epsilon_k \geq 0 \quad (22b)$$

$$\sum_{k=1}^{N-1} \epsilon_k \leq \phi(t) - f(x(t))_+ \quad (22c)$$

without altering feasibility and optimality of the solutions.

Moreover, in case f is given as the max of convex functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, n_f$, constraints (4e) and (22a) can be replaced by

$$f_i(x_N) \leq 0, \quad i = 1, \dots, n_f \quad (23a)$$

$$\epsilon_k \geq f_i(x_k), \quad i = 1, \dots, n_f, \quad k = 1, \dots, N-1 \quad (23b)$$

Similarly, if g is given as the max of convex functions $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, n_g$, (4d) can be replaced by

$$g_i(x_k, u_k) \leq 0, \quad i = 1, \dots, n_g, \quad k = 0, \dots, N-1 \quad (24)$$

The case $\ell_k = \max_{i=1, \dots, n_{\ell k}} \ell_{ik}$ can also be dealt with by introducing $\sum_{k=0}^N n_{\ell k}$ additional variables σ_{ik} , $k = 0, \dots, N$, and replacing (4a) with

$$\min \sum_{k=0}^N \sum_{i=1}^{n_{\ell k}} \sigma_{ik} \quad (25a)$$

$$\text{s.t. } \sigma_{ik} \geq \ell_{ik}(x_k, u_k), \quad i = 1, \dots, n_{\ell k}, \quad k = 0, \dots, N-1 \quad (25b)$$

$$\sigma_{iN} \geq \ell_{iN}(x_k), \quad i = 1, \dots, n_{\ell N} \quad (25c)$$

In conclusion, Algorithm 2 can be applied to solve (4) for any twice differentiable convex function f_i , g_i , ℓ_{ik} .

IV. CONSTRAINED TRACKING TO A SET

Consider an output vector

$$y(t) = Cx(t) \quad (26)$$

associated with process (8), with $y \in \mathbb{R}^p$, and a corresponding output reference $r \in \mathcal{R} \subset \mathbb{R}^p$, where \mathcal{R} is a polytope. Assume that the following linear system

$$\begin{aligned} 0 &= Ax_r + Bu_r - x_r \\ r &= Cx_r \end{aligned}$$

admits a unique solution (x_r, u_r) of steady-state state and input vectors for all $r \in \mathcal{R}$, and assume that \mathcal{R} is such that $g(x_r, u_r) < 0$.

We consider the problem of controlling (8) to a desired set $\mathcal{X}_T \triangleq \{x \in \mathbb{R}^n : S(x - x_r) \leq s\}$, around the equilibrium

state x_r , where $s > 0$, $s \in \mathbb{R}^{n_T}$, while satisfying the input constraints

$$u_{\min} \leq u(t) \leq u_{\max} \quad (27)$$

and the output constraints

$$y_{\min} \leq y(t) \leq y_{\max} \quad (28)$$

with $u_{\min} < 0 < u_{\max}$, $y_{\min} < 0 < y_{\max}$. In this case, we define the function g in (4d) as the convex and piecewise affine function

$$g(x, u) = \max_{i=1, \dots, q} \{G_i^x(Ax + Bu) + G_i^u u - g_i^0\} \quad (29)$$

where $G^x = \begin{bmatrix} 0 \\ I \\ -C \end{bmatrix}$, $G^u = \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix}$, $g^0 = \begin{bmatrix} u_{\max} \\ -u_{\min} \\ y_{\max} \\ -y_{\min} \end{bmatrix}$, and $q \triangleq 2m+2p$. An example of desired set \mathcal{X}_T is given by $S = \begin{bmatrix} C \\ -C \end{bmatrix}$, $s = \begin{bmatrix} e_{\max} \\ -e_{\min} \end{bmatrix}$, so that convergence to \mathcal{X}_T implies satisfying the constraint on the tracking error $e_{\min} \leq y - r \leq e_{\max}$ asymptotically.

A. Quadratic functions

We consider the ellipsoidal terminal set \mathcal{S} defined by

$$f(x) = (x - x_r)'P(x - x_r) - \rho_r \quad (30)$$

where $P = P' \geq 0$ and ρ_r are determined in accordance with the following theorem.

Theorem 6: Let $Q = Q' \geq 0$, $Q \in \mathbb{R}^{n \times n}$, $Y \in \mathbb{R}^{m \times n}$, $X = X' \geq 0$, $X \in \mathbb{R}^{m \times m}$, be the solution of the semidefinite program

$$\max (\det Q)^{\frac{1}{n}} \quad (31a)$$

$$\text{s.t. } \begin{bmatrix} Q & (AQ + BY)' \\ AQ + BY & \lambda Q \end{bmatrix} \geq 0 \quad (31b)$$

$$\begin{bmatrix} X & Y \\ Y' & Q \end{bmatrix} \geq 0 \quad (31c)$$

$$X_{ii} \leq \bar{u}_i^2, \quad i = 1, \dots, m \quad (31d)$$

$$\begin{bmatrix} Q & (AQ + BY)'C_i' \\ C_i(AQ + BY) & \bar{y}_i^2 \end{bmatrix} \quad (31e)$$

$$\begin{bmatrix} Q & Q'S_i' \\ S_i Q & S_i^2 \end{bmatrix}, \quad i = 1, \dots, n_T \quad (31f)$$

where $0 \leq \lambda \leq 1$ is a given contractive factor, and

$$\bar{u}_i \triangleq \min_{j=1, \dots, n_R} \{u_{\max, i} - (u_{r, j})_i, -u_{\min, i} + (u_{r, i})_i\} \quad (32a)$$

$$\bar{y}_i \triangleq \min_{j=1, \dots, n_R} \{y_{\max, i} - (r_j)_i, -y_{\min, i} + (r_j)_i\} \quad (32b)$$

for all $i = 1, \dots, q$, where $\{r_j\}_{j=1}^{n_R}$ are the vertices of \mathcal{R} . Then, by setting $K = YQ^{-1}$, $P \triangleq Q^{-1}$, $f(x) = (x - x_r)'P(x - x_r) - \rho_r$, the set $\mathcal{S} = \{x : (x - x_r)'P(x - x_r) \leq \rho_r\}$ is constrained controlled invariant under the control law $u = K(x - x_r) + u_r$ in that $x \in \mathcal{S}$ implies

$$Ax + Bu \in \mathcal{S} \quad (33a)$$

$$u_{\min} \leq K(x - x_r) + u_r \leq u_{\max} \quad (33b)$$

$$y_{\min} \leq C(Ax + Bu) \leq y_{\max} \quad (33c)$$

$$Sx \leq s \quad (33d)$$

for all $r \in \mathcal{R}$, where

$$\rho_r = \min_{i=1,\dots,q+n_T} \left\{ \frac{\bar{b}_i^2}{\bar{A}_i Q \bar{A}_i'} \right\} \geq 1 \quad (34a)$$

$$\bar{A} = \begin{bmatrix} K & & \\ & -K & \\ C(A+BK) & & \\ -C(A+BK) & & \\ S & & \end{bmatrix}, \quad \bar{b} = \begin{bmatrix} u_{\max} - u_r \\ -u_{\min} + u_r \\ y_{\max} - r \\ -y_{\min} + r \\ s \end{bmatrix} \quad (34b)$$

Proof: Let $\Delta x \triangleq x - x_r$, $\Delta u \triangleq u - u_r$, $\Delta y \triangleq y - r$. Clearly $\Delta x(t+1) = Ax + Bu - x_r = A\Delta x(t) + B\Delta u(t)$ and $\Delta y(t) = C\Delta x(t)$, along with the constraints $u_{\min} - u_r \leq \Delta u(t) \leq u_{\max} - u_r$, $y_{\min} - r \leq C\Delta x(t) \leq y_{\max} - r$. The robust satisfaction of properties (33) with respect to $r \in \mathcal{R}$ for all x such that $\Delta x' P \Delta x \leq 1$, under the control law $\Delta u(t) = K\Delta x(t)$, follows by standard arguments from the inequality constraints in (31) (see, e.g., [23]). For a given $r \in \mathcal{R}$, the scalar ρ_r defined by (34) provides the largest ellipsoid centered in x_r and defined by P such that (33) are satisfied, where $\rho_r \geq 1$. ■

B. Polyhedral terminal set \mathcal{S}

For a given under asymptotically stabilizing feedback control law $u(t) = K(x(t) - x_r) + u_r$, consider the polyhedral terminal set \mathcal{S} defined by

$$f(x) = \max\{H_i'(x - x_r) - K_i\} \quad (35)$$

where $\mathcal{S} = \{x : H(x - x_r) \leq K\} = \{\Delta x : H\Delta \leq K\}$ is a maximum admissible polyhedral invariant set [24] for the closed-loop system $\Delta x(t+1) = (A+BK)\Delta x(t)$ and with respect to the constraints $\bar{A}\Delta x \leq \bar{b}_{\min}$, where \bar{A} is defined in (34b), and \bar{b}_{\min} is defined as in (34b) by replacing $(u_{\max} - u_r)_i$ with $\min_{j=1,\dots,n_R}\{(u_{\max} - u_{r_j})_i\}$, $(-u_{\min} + u_r)_i$ with $\min_{j=1,\dots,n_R}\{(-u_{\min} + u_{r_j})_i\}$, $i = 1, \dots, m$, $(y_{\max} - r)_i$ with $\min_{j=1,\dots,n_R}\{(y_{\max} - r_j)_i\}$ and $(-y_{\min} + r)_i$ with $\min_{j=1,\dots,n_R}\{(-y_{\min} + r_j)_i\}$, $i = 1, \dots, p$. Clearly, the size of \mathcal{S} depends on the size of \mathcal{X}_T and on how large are the boxes $\{y \in \mathbb{R}^p : y_{\min} \leq y \leq y_{\max}\}$ with respect to \mathcal{R} and $\{u \in \mathbb{R}^m : u_{\min} \leq u \leq u_{\max}\}$ with respect to the set $\{u \in \mathbb{R}^m : u = u_r, r \in \mathcal{R}\}$.

V. SIMULATION RESULTS

Consider the linear system described by the transfer function

$$G(s) = \frac{26(s+1)}{s^2 + 2s + 26} \quad (36)$$

Model (36) is converted to discrete-time by exact sampling plus zero-order hold with sampling time $T_s = 0.2$ s, resulting in the state-space model with matrices $A = \begin{bmatrix} 0.4424 & 1 \\ -0.4746 & 0.4424 \end{bmatrix}$, $B = \begin{bmatrix} 2.0623 \\ 0 \end{bmatrix}$, $C = [-0.7013 \ 1.9407]$, $D = 0$. The system is subject to the constraints

$$-1 \leq u \leq 1, \quad -1 \leq y \leq 1 \quad (37)$$

leading to defining $g(x, u)$ as in (29). We setup the MPC problem (4) with $N = 6$,

$$\ell_k(x, u) = (x - x_r)'Q(x - x_r) + (u - u_r)'R(u - u_r) \quad (38)$$

$Q = 10C'C$, $R = 1$, for all $k = 0, \dots, N-1$, and

$$\ell_N(x) = (x - x_r)'P(x - x_r) \quad (39)$$

where P is the solution of the discrete algebraic Riccati equation associated with A , B , Q , and R , and $x_r = \begin{bmatrix} 2.6252 \\ 1.4639 \end{bmatrix} r$, $u_r = r$. The possible reference signals are restricted in the interval $\mathcal{R} = [-0.9, 0.9]$, while the desired target set $\mathcal{X}_T = \{x \in \mathbb{R}^2 : \|x - x_r\|_{\infty} \leq 0.1\}$. We start from the initial condition $x(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and command the set-point $r = 0.5$. We consider the cumulated cost

$$J \triangleq \sum_{t=0}^{10} (\ell_N(x_N^t) + \sum_{k=0}^{N-1} \ell_k(x_k^t, u_k^t))$$

as a measure of closed-loop performance, where $\{u_k^t\}_{k=0}^{N-1}$ is the solution of problem (4) chosen at time t , and $\{x_k^t\}_{k=0}^N$ the corresponding state trajectory. We consider two settings for function f :

Case (i): the convex quadratic function as in (30), where $P = \begin{bmatrix} 105.4493 & 23.9713 \\ 23.9713 & 105.4493 \end{bmatrix}$ is obtained by (31) with $\lambda = 1$, along with the terminal gain $K = [0.1968 \ -0.2898]$. Table I shows the cumulated cost J for different maximum values of permitted CPU time to solve problem (4). The corresponding trajectories are depicted in Figures 1, 2.

TABLE I
PERFORMANCE VS. ALLOCATED CPU TIME (ELLIPSOIDAL CONSTRAINTS)

max CPU time (ms)	Cumulated cost J
unbounded	8.7865
60	22.6572
40	26.4663
20	31.4528

Case (ii): the convex piecewise affine function as in (35), where $H = \begin{bmatrix} 1 & 0 & -1 & 0 & 0.4424 & -0.4424 \\ 0 & 1 & 0 & 0 & 1 & -1 \end{bmatrix}'$, $K = 0.1 [1 \ 1 \ 1 \ 1 \ 1 \ 1]'$ is the maximum λ -contractive invariant set for the closed-loop system $\Delta x(t+1) = (A+BK_{\text{LQR}})\Delta x(t)$, K_{LQR} is the LQR gain associated with A , B , Q , and R , and $\lambda = 1$, computed as described in [25]. Table II shows the cumulated cost J for different maximum values of permitted CPU time to solve problem (4). The corresponding trajectories are depicted in Figures 3, 4.

TABLE II
PERFORMANCE VS. ALLOCATED CPU TIME (POLYHEDRAL CONSTRAINTS)

max CPU time (ms)	Cumulated cost J
unbounded	9.0075
10	23.4491
5	26.4646
1	26.7551

Finally, we compare the performance of the new solver described in Section III (implemented in interpreted MATLAB code) against the commercial solver Gurobi 5.6.2 [26] in solving QCQP and QP problems to optimality, and also to qpOASES [6] for QP's. We consider problems deriving from the MPC setup with ellipsoidal (QCQP) and polyhedral (QP) constraints described above, for an increasing prediction horizon N . The results are depicted in Figure 5 (QCQP case) and Figure 6 (QP case), respectively.

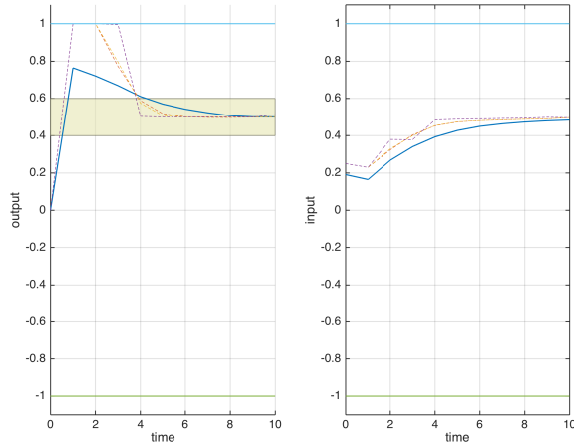


Fig. 1. System trajectories for varying values of available CPU time (solid=optimal), ellipsoidal constraints

VI. CONCLUSIONS

The contribution of this paper is twofold. From an optimization viewpoint, we have introduced a very efficient numerical solver that can solve convex feasibility and optimization problems, and that is at least an order of magnitude faster than commercial state-of-the-art (interior-point) solvers as the dimension of the problem increases. By taking advantage of the way the solver computes an optimal solution via a sequence of convex feasibility problems, from a control viewpoint we proposed an MPC strategy for convergence to a terminal set that allows an *anytime optimization* philosophy, that is of improving the optimality of the control move with respect to a given performance specification only if CPU resources are available during the sampling interval. We believe that the approach has potential applications in embedded MPC systems where a large-enough time-slot for computations cannot be guaranteed a priori, a rather typical situation in multitask real-time systems.

REFERENCES

- [1] D. Mayne and J. Rawlings, *Model Predictive Control: Theory and Design*. Madison, WI: Nob Hill Publishing, LCC, 2009.
- [2] J. Maciejowski, *Predictive Control with Constraints*. Harlow, UK: Prentice Hall, 2002.
- [3] A. Bemporad, "Model-based predictive control design: New trends and tools," in *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, CA, 2006, pp. 6678–6683.
- [4] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.
- [5] N. Ricker, "Use of quadratic programming for constrained internal model control," *Ind. Eng. Chem. Process Des. Dev.*, vol. 24, no. 4, pp. 925–936, 1985.
- [6] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.
- [7] C. Schmid and L. Biegler, "Quadratic programming methods for reduced Hessian SQP," *Computers & Chemical Engineering*, vol. 18, no. 9, pp. 817–832, 1994.
- [8] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, pp. 1–27, 2010.

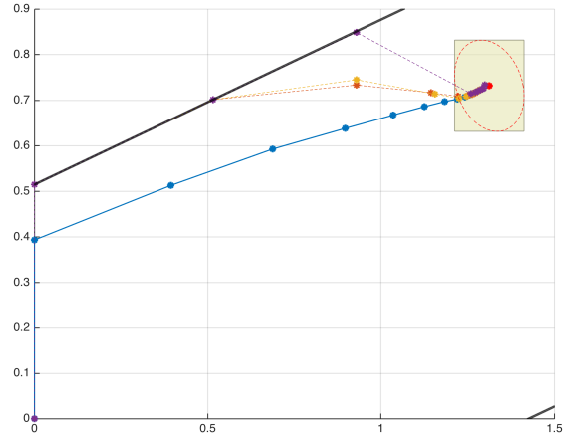


Fig. 2. State-space trajectories for varying values of available CPU time (solid=optimal), ellipsoidal constraints

- [9] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Contr. Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.
- [10] M. Zeilinger, D. Raimondo, A. Domahidi, M. Morari, and C. Jones, "On real-time robust model predictive control," *Automatica*, vol. 50, pp. 683–694, 2014.
- [11] P. Patrinos, P. Sopasakis, and H. Sarimveis, "A global piecewise smooth Newton method for fast large-scale model predictive control," *Automatica*, vol. 47, no. 9, pp. 2016–2022, 2011.
- [12] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.
- [13] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Trans. Automatic Control*, vol. 59, no. 1, pp. 18–33, 2014.
- [14] M. Rubagotti, P. Patrinos, and A. Bemporad, "Stabilizing embedded MPC with computational complexity guarantees," in *Proc. European Control Conf.*, Zürich, CH, 2013, pp. 3065–3070.
- [15] D. Fontanelli, L. Greco, and A. Bicchi, "Anytime control algorithms for embedded real-time systems," in *Hybrid Systems: Computation and Control*, M. Egerstedt and B. Mishra, Eds. Springer-Verlag, 2008, pp. 158–171.
- [16] A. Bemporad, "A predictive controller with artificial Lyapunov function for linear systems with input/state constraints," *Automatica*, vol. 34, no. 10, pp. 1255–1260, 1998.
- [17] D. Bernardini and A. Bemporad, "Stabilizing model predictive control of stochastic constrained linear systems," *IEEE Trans. Automatic Control*, vol. 57, no. 6, pp. 1468–1480, 2012.
- [18] P. Scokaert, D. Mayne, and J. Rawlings, "Suboptimal model predictive control (feasibility implies stability)," *IEEE Trans. Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.
- [19] S. D. Cairano and F. Borrelli, "Constrained tracking with guaranteed error bounds," in *Proc. 52nd IEEE Conf. on Decision and Control*, Florence, Italy, 2013, pp. 3800–3805.
- [20] P. Patrinos and A. Bemporad, "Proximal newton methods for convex composite optimization," in *Proc. 52nd IEEE Conf. on Decision and Control*, Florence, Italy, 2013, pp. 2358–2363.
- [21] P. Patrinos, L. Stella, and A. Bemporad, "Forward-Backward truncated Newton methods for convex composite optimization," 2014, submitted for publication. <http://arxiv-web3.library.cornell.edu/abs/1402.6655>.
- [22] D. Bertsekas, "A note on error bounds for convex and nonconvex programs," *Computational Optimization and Applications*, vol. 12, no. 1-3, pp. 41–51, 1999.
- [23] M. Kothare, V. Balakrishnan, and M. Morari, "Robust constrained model predictive control using linear matrix inequalities," *Automatica*, vol. 32, no. 10, pp. 1361–1379, 1996.
- [24] E. Gilbert and K. T. Tan, "Linear systems with state and control constraints: the theory and applications of maximal output admissible

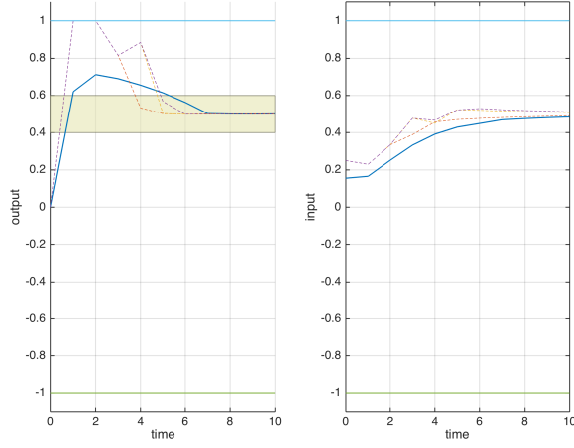


Fig. 3. System trajectories for varying values of available CPU time (solid=optimal), polyhedral constraints

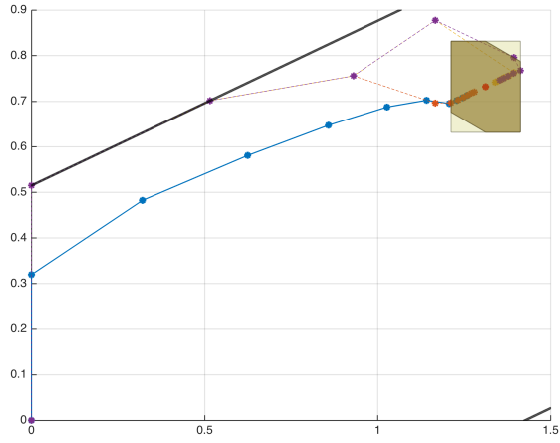


Fig. 4. State-space trajectories for varying values of available CPU time (boldface=optimal), polyhedral constraints

sets,” *IEEE Trans. Automatic Control*, vol. 36, no. 9, pp. 1008–1020, 1991.

[25] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, “Ultra-fast stabilizing model predictive control via canonical piecewise affine approximations,” *IEEE Trans. Automatic Control*, vol. 56, no. 12, pp. 2883–2897, 2011.

[26] Gurobi Optimization, Inc., *Gurobi Optimizer Reference Manual*, 2014. [Online]. Available: <http://www.gurobi.com>

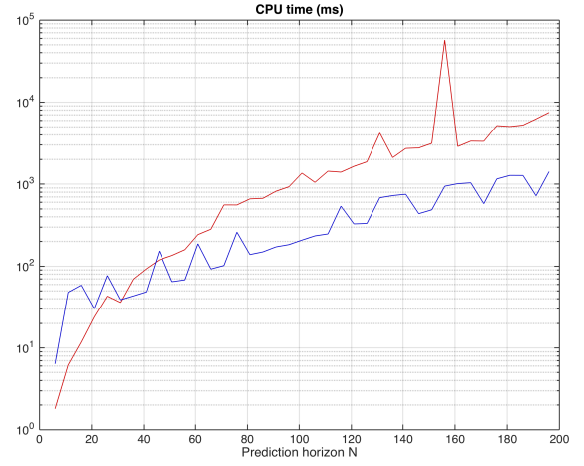


Fig. 5. QCQP: Algorithm 2 (blue) vs Gurobi (red)

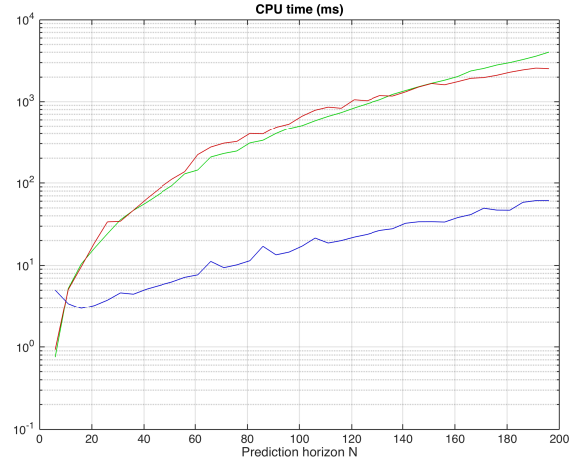


Fig. 6. QP: Algorithm 2 (blue), Gurobi (red), qpOASES (green)